A self-correcting neighbor protocol for mobile ad-hoc wireless networks

Marc Mosko and J.J. Garcia-Luna-Aceves
University of California at Santa Cruz
Baskin Engineering
Santa Cruz, CA 95064, USA
{mmosko, jj}@cse.ucsc.edu

Abstract— **Mobile wireless ad-hoc networks lack some basic abilities taken for granted in wired networks, such as the ability to know adjacent nodes. We present a neighbor discovery protocol, with particular application to broadcast flooding. The Neighbor Exchange Protocol (NXP) has two main improvements over simple periodic broadcast schemes: (1) it only sends Hello packets when necessary to maintain topology and (2) uses sequence numbers in redistributed information to aid in convergence. In simulation, we compare NXP to a periodic protocol and simple flooding for all-node packet broadcasts and two dissemination techniques. We show that we maintain similar delivery rates while using fewer control packets in most configurations.**

I. Introduction

Wireless ad-hoc networks are characterized by a radio channel and arbitrary topology without fixed infrastructure, such as cell sites or base stations. All such networks need a method to detect and organize nodes, whether it is to compute a schedule for TDMA-style MAC layers, determine adjacencies for routing, or perform what is known as dominating-set routing [1], which uses localized neighborhood information to create packet distribution backbones.

Neighbor protocols are designed to exchange node information for determining which nodes are "alive" and reachable. They generally fall in to two groups: periodic or event-based. Periodic protocols broadcast Hello packets with some, possibly variable, frequency. The frequency may vary based on network load, node mobility, or relative group mobility. Event-based schemes send Hello packets based on events such as distance moved or detected topology changes.

Many protocols use neighbor information to control broadcast distribution over ad-hoc networks. Some use periodic Hello messages [1], [2], [3], [4], [5] and others use event-based updates [6], [7], [8]. Periodic schemes usually exchange limited information – such as only node ids – over a limited distance. Hello packets in location-based schemes broadcast the sending node's location, such as determined by GPS. The location-based protocols generally serve as full routing protocols, and are not limited to two-hop neighbor information. The location packets may flood the network.

This work was supported in part by the Office of Naval Research (ONR) under grant N00014-99-1-0167.

The present work does not address the differences between periodic topology protocols and location-based protocols. Location-based protocols have a significantly different approach to connectivity and routing. NXP is suited towards distance-vector routing protocols and connectivity-based dominating set broadcast distribution.

Broadcast distribution control generally follows a type of dominating-set routing or re-broadcasting based on geometry. Neighbor information controls how nodes propagate packets – either repeating a packet or suppressing a packet. The protocols assume full duplex links. If node A can hear node B's Hello packet, then node B may hear A. In fact, this is sometimes not the case in common-channel wireless networks, where asymmetric fading and location-dependent noise levels may favor hearing one node or another.

A problem arises when a node makes forwarding decisions based on incorrect information. If in the above example, A actually cannot hear B, but B hears and advertises A as a one-hop neighbor, some third node C may choose B to forward to A when in fact there may be a better forwarder. Section II describes the Neighbor Exchange Protocol (NXP), which addresses this weakness by including per-node sequence numbers in topology broadcasts and using a three-state machine – Up, Hold, and Down – where nodes in the Hold state have not been full-duplex verified.

NXP converges to correct topology information and maintains liveliness in a mobile environment without requiring Hello packets be sent on a fixed schedule. It also allows using small KeepAlive packets rather than full topology Hello datagrams. The key feature of NXP is to advertise neighbors' ids, state (up or hold), and sequence number. By repeating the most recently heard sequence numbers for each one-hop neighbor, NXP can detect inconsistencies and pro-actively try to correct them. Including a node state allows advertising unverified nodes (nodes in simplex state) and also serves to indicate to one-hop neighbors if they are going to be aged-out because of lost neighbor packets.

Topology Broadcast Reverse Path Forwarding (TBRPF) [9] uses the TBRPF Neighbor Discovery (TND) mechanism. TND has several similarities to NXP. As of draft 3 (current draft is 5), TND is modular and independent of TBRPF, similar to how we have used an external NXP process. TND uses differential Hello messages whereas NXP uses either com-

Report Documentation Page						*Form Approved* *OMB No. 0704-0188*
Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.						
1. REPORT DATE **2002**		2. REPORT TYPE			3. DATES COVERED **00-00-2002 to 00-00-2002**	
4. TITLE AND SUBTITLE **A self-correcting neighbor protocol for mobile ad-hoc wireless networks**					5a. CONTRACT NUMBER	
					5b. GRANT NUMBER	
					5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S)					5d. PROJECT NUMBER	
					5e. TASK NUMBER	
					5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of California at Santa Cruz,Department of Computer Engineering,Santa Cruz,CA,95064**					8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)					10. SPONSOR/MONITOR’S ACRONYM(S)	
					11. SPONSOR/MONITOR’S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release; distribution unlimited**						
13. SUPPLEMENTARY NOTES						
14. ABSTRACT						
15. SUBJECT TERMS						
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON	
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**5**		

plete one-hop neighbor information Hellos or Keepalives which only carry a sequence number. TND sends a Hello every HELLO_INTERVAL, whereas NXP will not send a Hello if the MAC layer has otherwise sent a packet and there has not been some other event that mandates a Hello packet. NXP relies on promiscuous mode operation.

TND maintains symmetric links by repeating recently heard Hellos in the NEIGHBOR_REQUEST field. A receiver scans this list to see if it is listed and if not, it sends Hellos until a 2-way link is established, which is similar to NXP's "force hello" mechanism, but NXP also considers sequence number differences. NXP extends this idea to all common one-hop neighbors and uses Poll packets to converge topology information.

TND makes uses of sequence number differences to detect the number of lost Hellos and if it is greater than a threshold a pair of nodes may agree to ignore each other by declaring the link "lost". NXP does not have any such mechanism, but it does detect intermittent links by redistributing all sequence numbers of neighboring nodes and comparing the resulting topologies, and may try to converge the topology with Poll packets.

Section III presents the results of simulation experiments using three broadcast distribution protocols over two neighbor protocols and flooding. The broadcast distribution protocols are AHBP and AHBP-EX [5], and Dominant Pruning (DP) [3]. The analysis considered three graph densities, four traffic rates, two mobility rates, and five Hello periods. Except for one combination, NXP performed better than a periodic protocol for dominating-set broadcast distribution. NXP either had a higher delivery ratio and a higher efficiency, or a higher delivery ratio. Flooding had the highest overall delivery ratio in all simulations, but generally was very inefficient in terms of the ratio of packets transmitted to packets received. The exception is AHBP-EX using a periodic protocol with a long Hello period. It appears that the out-dated neighbor information gives AHBP-EX a boost: stale neighbor information allows AHBP-EX to flood more often and results in higher delivery rates.

II. Protocol

We first present a general description of NXP then provide pseudo-code of the principle routines. This section focuses on the use of sequence numbers to aid in direct and indirect topology convergence and the state changes between Up, Hold, and Down for neighboring nodes.

The protocol works by combining pro-active Hello packets that contain one-hop topology information with MAC layer snooping. Nodes only send Hello packets if the MAC layer has not transmitted any other packet within a timeout period. In an 802.11-style MAC layer, all packets including control packets contain the source address and may serve as keep-alive indicators to neighboring nodes.

Our current implementation uses a fixed beaconing interval in which at least one packet (user data, NXP Hello, NXP Poll, or NXP Keepalive) must be sent. It would be possible to use variable frequency periods or event-based triggers, such as mobility rate or distance moved, in other instances of NXP. The present work focuses on the redistribution of sequence numbers and indirect detection of topology changes using those numbers.

NXP uses three types of packets and four timers. The packets are Hello, KeepAlive, and Poll. The timers are SendMessage, Purge, Recompute, and Poll. We refer to the length of the mean SendMessage timer as Period. A node sets timers with a jitter variation chosen uniformly over an interval. NXP also responds to the event Message: the reception of a packet from the MAC layer. Receiving a packet may promote neighbor status from Unknown to Up, Down to Hold, or Hold to Up. Periodic Purge timers may demote status from Up to Hold or Hold to Down.

NXP maintains two tables. The NbrTable stores a row for each one-hop neighbor, described below. NXP passes this information to a topology algorithm on topology changes, rate limited by the Recompute timer. The NbrPoll table lists nodes in an inconsistent sequence-number state. A node will periodically send a Poll packet to these nodes, gated by the Poll timer.

Each node has a HelloSeqno, which is monotonically increasing for each Hello sent. A KeepAlive packet includes the current HelloSeqno, but does not increment the value. Poll messages do not include a sequence number. We do not currently consider issues of sequence number wrap-around and node failures resulting in reusing sequence numbers.

A neighbor node may be in one of three states: Up, Hold, Down. Hello packets advertise both Up and Hold nodes, indicating the state. A receiver only uses Up nodes in topology computations. A node in the Down state is considered disconnected and all topology information from it is discarded. A node in the Hold state is in a hysteresis between Up and Down.

A. Data Structures

The NbrPoll table is a simple list of node addresses. All nodes listed are candidates for a Poll message when the Poll timer expires. The NbrTable stores records consisting of {id, lastHeard, lastHelloSeqno, state, heardCount, nbr}. `lastHeard` tracks the arrival time of the last packet from a given one-hop neighbor. `lastHelloSeqno` tracks the last valid HelloSeqno from the node. `state` = {Up,Hold,Down} and `heardCount` tracks the number of packets heard since the last Purge.

A Hello packet sent to the MAC layer has the fields {type, seqno, size, nodeList, stateList, seqnoList}. The field `type` is {Hello,KeepAlive,Poll}, indicating the type of packet. The field `seqNo` is the sending node's current HelloSeqno. When sending a Hello packet, the source increments `HelloSeqno` before sending the packet. For KeepAlive packets, a source uses its current `HelloSeqno`. Poll packets do not use `seqno`. The first Hello sent by a node has seqNo=1. `size` is the number of entries in the lists `nodeList, stateList, seqnoList`. `nodeList` lists all one-hop neighbors in the Up or Hold state. For each address $A \in nodeList$, `seqnoList[A]` contains the corresponding HelloSeqno known at S for A. This entry may be 0, which indicates the sender has not received a Hello from A, but only snooped a packet. `stateList[A]` likewise is the state for A.

A KeepAlive packet has only the `type` and `seqNo` fields. It does not repeat topology information. A Poll packet is the same as a Hello packet, but the `type` indicates it is a Poll and it does not use `seqnoList` or `stateList`. The nodes listed in `nodeList` are requested to respond with a Hello packet at the expiration of their next SendMessage timer.

B. Node Initialization

At initialization, a node creates empty NbrTable and NbrPoll tables. It sets the three timers SendMessage, Purge, and Poll. The Recompute timer is only set by protocol events. There are three global variables, set as: `forceHello` = true, `recomputePending` = false, `helloSeqno` = 0. NXP also keeps track of the number of packets transmitted by the MAC. We initialize `macUnicast`, `macBroadcast`, `macMulticast`, and `macControl` to zero.

NXP uses the constants HoldDown and AgeOut to determine state changes. HoldDown controls when an Up neighbor becomes a Hold neighbor. We use 1 * (Period + max jitter). AgeOut controls when a Hold neighbor becomes a Down neighbor. We use 3 * (Period + max jitter). Using a longer HoldDown results in significantly less overhead and somewhat lower delivery rates. The constant HoldCnt determines the number of consecutive packets needed to transition a node from Hold to Up. We use HoldCnt=2.

C. Algorithms

NXP's three principle routines are `ProcessMessage()`, `Timer()`, and `ProcessHello()`. The environment calls `ProcessMessage()` on the arrival of any packet for NXP. These include packets addressed to the NXP SAPI and to packets snooped by the MAC layer. The environment calls `Timer()` at the expiration of any of the four times. `ProcesssHello()` is called by `ProcessMessage()` for Hello PDUs.

The salient features of `Timer()` is that a SendMessage timer will only send a packet if `forceHello` is true or if the MAC layer statistics show that the MAC layer has not sent any packets since the last check. NXP compensates for NXP-originated packets and does not count those. A Purge timer will only force a topology recompute when a node transitions from Hold to Down. Up to Hold transitions do not cause a recompute.

`ProcessMessage()` and `ProcessHello()` contain the code to pro-actively use Poll and Hello messages in response to topology changes detected through sequence numbers. In lines 8 – 10 of `ProcessMessage()`, the receiving node checks that the sequence number of a KeepAlive packet is the same as the stored `node.lastHeardSeqno`. If it is newer, then `node` will queue a Poll for `msg.source`. In lines 14 – 17, a receiving node will queue a Poll packet if it receives a packet from an unknown node and the packet is not a Hello.

`ProcessHello()` performs indirect detection of topology changes in lines 14 – 16. For each node $j \in msg.data$ listed in a received Hello packet, the receiver checks to see if j is a known one-hop neighbor. If so, the receiver compares the stored `lastHeardSeqno` against the redistributed `msg.seqnoList`. If the received Hello has a more current sequence number, the receiver has out-of-date information and will queue a Poll request for node j.

```
PROCESSMESSAGE($node, msg$)
(1)   $isnew$ ←**false**
(2)   $row \leftarrow node.NbrTable[msg.source]$
(3)   **if** $row$ is NIL
(4)       Create new row in Up state
(5)       $isnew$ ←**true**
(6)   **if** $msg$ is Hello
(7)       PROCESSHELLO($node, msg, row$)
(8)   **else if** $msg$ is Keepalive
(9)       **if** $msg.seqno > row.lastHeardSeqno$
(10)          Add $msg.source$ to node.NbrPoll
(11)  **else if** $msg$ is Poll
(12)      **if** $node.id \in msg.data$
(13)          $node.forceHello$ ←**true**
(14)  **else**
(15)      Promiscuous snooped packet
(16)      **if** $isnew$ = **true**
(17)          Add $msg.source$ to node.NbrPoll
(18)  **if** $isnew$ = **true**
(19)      FORCERECOMPUTE(node)
(20)  **else**
(21)      **if** $row.state$ = Up
(22)          $row.lastHeard$ ←now
(23)      **else if** $row.state$ = Hold
(24)          **if** $now - row.lastHeard \leq period + jitter$
(25)              $row.heardCount++$
(26)          **else**
(27)              $row.heardCount$ ←1
(28)          **if** $row.heardCount \geq holdcnt$
(29)              $row.state$ ←Up
(30)          $row.lastHeard$ ←now
(31)      **else**
(32)          Packet from down node
(33)          $row.heardCount$ ←1
(34)          $row.lastHeard$ ←now
(35)          $row.state$ ←Hold
```

III. SIMULATION

We conducted several types of simulations using GloMoSim [10] with an 802.11 MAC layer with a 250m transmission range. The simulations measured the delivery ratio of packets received to packets sent and the number of neighbor over-head packets. We simulated a wide variety of conditions, which show that NXP has the same delivery ratio as a periodic protocol, but with lower overhead.

In each scenario, there are four source nodes, which send 500 byte UDP packets from a Poisson source with mean rate of 2,

```
TIMER($node, timer$)
(1)  if $timer$ is Recompute
(2)      Send all up node information in $node.NbrTable$ to topology algorithm.
(3)  else if $timer$ is SendMessage
(4)      if ($node.forceHello$ = true ) or (MAC layer quiet)
(5)          $hello.type = hello$
(6)          $node.seqno++$
(7)          $hello.seqno = node.seqno$
(8)          For each $\{id, state, seqno\} \in node.NbrTable$ where state is Up or Hold, add to $hello.data$.
(9)          Transmit hello.
(10)     Set SendMessage timer
(11) else if $timer$ is Poll
(12)     $PollList$ ←NIL
(13)     foreach $row \in node.NbrPoll$
(14)         if $row.state \neq Down$
(15)             Add $row.id$ to $PollList$
(16)     if $PollList$ not NIL
(17)         Create Hello packet with $hello.data \leftarrow \{PollList, 0, 0\}$ and $hello.seqno \leftarrow 0$. Transmit.
(18)     Set Poll timer
(19) else if $timer$ is Purge
(20)     foreach $row \in node.NbrTable$
(21)         if $row.state$ = Up
(22)             if $now - row.lastHeard \geq HoldDown$
(23)                 $row.state$ ←Hold
(24)                 $row.heardCount$ ←0
(25)         else if $row.state$ = Hold
(26)             if $now - row.lastHeard \geq Ageout$
(27)                 $row.state$ ←Down
(28)                 $row.heardCount$ ←0
(29)                 $row.nbrs$ ←NIL
(30)                 FORCERECOMPUTE($node$)
```

```
PROCESSHELLO($node, msg, row$)
(1)  if $msg.seqno > row.lastHeardSeqno$
(2)      $oldnbrs \leftarrow row.nbrs$
(3)      $row.nbrs$ ←NIL
(4)      $foundme$ ←false
(5)      $row.lastHeardSeqno \leftarrow msg.seqno$
(6)      foreach $\{n, state, seq\} \in msg.data$
(7)          if $state$ = Up
(8)              Add $n$ to $row.nbrs$ list
(9)          if $n = node.id$
(10)             $foundme$ ←true
(11)             if ($seq < node.seqno$) or ($state$ = Hold)
(12)                 $node.forceHello$ ←true
(13)         else
(14)             $row2 \leftarrow node.NbrTable[n]$
(15)             if ($row2$ not NIL) and ($seqno > row2.lastHeardSeqno$)
(16)                 Add node $n$ to $node.NbrPoll$
(17)     if $foundme$ = false
(18)         $node.forcehello$ ←true
(19)     if $row.nbrs \neq oldnbrs$
(20)         FORCERECOMPUTE($node$)
```

Description	Values
Hello period (sec)	0.5, 1, 2, 4, 8
Distribution	Flood, DP, AHBP, AHBP-EX
Neighbor Proto	None, Periodic, NXP
Source rate (pps)	2, 10, 20, 40
Mobility	1-10 m/s or 5-20 m/s
Pattern	random waypoint, 0 sec pause
Sim time	300 sec

TABLE I
SIMULATION VARIABLES

10, 20, and 40 packets per second per node, depending on the simulation scenario. Nodes 1 and 2 were on from [0s - 60s] and [120s- 180s]. Nodes 3 and 4 were on from [30s - 90s] and [150s - 210s]. Repeater nodes add a random exponential delay when repeating packets, with mean delay of 5 milliseconds.

There are five general scenarios: base case (BC), high density (HD), high mobility (HM), group mobility (GM), and low density (LD), all being variations on parameters listed in Table I. The BC scenario uses 50 nodes with velocities 1 – 10 m/s placed at random in the simulation space. The HD scenario uses 100 nodes. The HM scenario uses 50 nodes with velocities 5 – 20 m/s. The GM scenario is BC with a circularly symmetric node distribution of four rings with radii of 50m, 150m, 400m, and 750m. There were 5, 10, 16, and 19 nodes in each ring, respectively. The GM scenarios model high noise/low noise environments by grouping nodes. The LD scenario is BC, but with 25 nodes.

Each scenario is run with 123 configurations, based on variations of Table I. The Hello period is the setting for the SendMessage timer. The Purge timer is set to the same mean as the SendMessage timer. The Poll timer mean is 1.5 times longer than the SendMessage timer. The Recompute timer is $1/20^{th}$ the SendMessage timer. The jitter in timer all timers except Recompute is $\pm 1/5^{th}$ the SendMessage length. The Recompute timer is deterministic. The Distribution methods are Distributed Pruning (DP) [3], Ad-hoc Broadcast Protocol (AHBP) [5], and blind flooding with packet cache (Flood). In the simulation runs, we denote AHBP as AH and AHBP-EX as AX. The Neighbor Protocols are Periodic (PR) and NXP (NX). Note that flooding does not use a neighbor protocol and does not depend on Hello periods.

We have summarized the behavior of broadcast flooding in to two metrics with the independent variable being the Hello period. The Delivery Ratio is the total number of unique packets received by all nodes divided by the total number of packets originated by all nodes. The Inefficiency is the total number of packets transmitted by all nodes divided by the total number of unique packets received by all nodes normalized by (divided by) the total number of packets originated by all sources. In computing the averages, we consider a source node originating packets to receive all those packets

Based on our simulation parameters, flooding always had the best delivery ratio. Fig. 1 plots the Delivery Ratio for the five Hello periods. As the Hello period increases, the delivery ratio

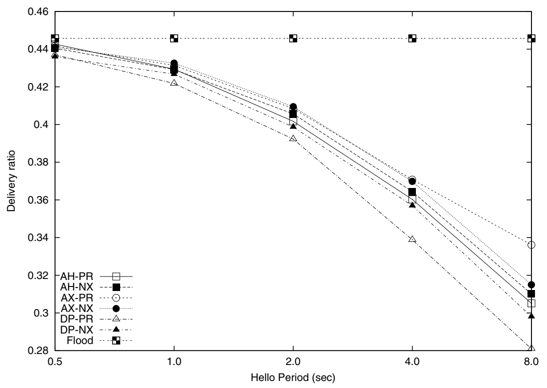

Fig. 1. Delivery ratio

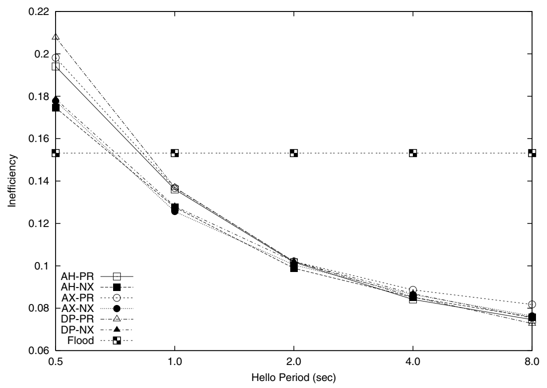

Fig. 2. Inefficiency

of the dominating-set distribution protocols decreases. Flooding is invariant to Hello period as it does not use a neighbor protocol. At long Hello periods (8 seconds), AHBP-EX with a Periodic protocol has a higher delivery ratio. Stale one-hop information allows AHBP-EX to flood more packets resulting in a higher delivery ratio.

In terms of efficiency (inefficiency in our plots), Fig. 2 shows that for very short Hello periods, dominating-set distribution with neighbor protocols is less efficient than flooding and from Fig. 1 has a lower delivery rate. For Hello periods of 1 second or longer, the adaptive protocols become more efficient than Flooding. A 1 second Hello period has almost the same delivery ratio as flooding at a higher efficiency.

The absolute values of results vary considerably depending on the scenarios (BC, GM, HD, HM, LD), but the general trends are similar. The notable exception is that in the HD and GM configurations, dominating-set distribution have higher delivery ratios than flooding for Hello periods of 2 seconds and under for high packet rates (more than 10 pps).

IV. Conclusion

We have presented the Neighbor Exchange Protocol (NXP), which uses variable frequency Hello packets to maintain two-hop topology information in an ad-hoc network. NXP also uses small KeepAlive packets rather than rebroadcast full topology Hello packets if there has been no change to topology within a timeout period. By redistributing sequence numbers and node states in Hello packets, NXP may detect topology inconsistencies.

NXP uses and redistributes sequence numbers and has a three state (up, hold, down) neighbor model. This allows direct detection of inconsistencies. A Poll mechanism allows nodes to request full-topology Hello packets when needed.

Simulation experiments showed that NXP yields delivery ratios that are in most cases as good or better than those attained by a periodic protocol. Where the delivery ratio is lower, NXP also has a higher efficiency. An optimization to the protocol would be to piggyback Poll packets on Hello/Keepalive packets, which would further reduce the protocol overhead.

References

[1] Jie Wu, Ming Gao, and I. Stojmenovic, "On calculating power-aware connected dominating sets for efficient routing in ad hoc wireless networks," in *Proc. International Conf. Parallel Processing*, Sept. 2001, pp. 346–54.

[2] Sze-Yao Ni, Yu-Chee Tseng, Yuh-Shyan Chen, and Jang-Ping Sheu, "The broadcast storm problem in a mobile ad hoc network," *MobiCom'99*, Aug 1999.

[3] H. Lim and C. Kim, "Flooding in wireless ad hoc networks," *Computer Communications*, vol. 24, no. 3-4, pp. 353 – 363, Feb 2001.

[4] Wei Peng and Xi-Cheng Lu, "On the reduction of broadcast redundancy in mobile ad hoc networks," in *2000 First Annual Workshop on Mobile and Ad Hoc Networking and Computing MobiHOC*. Sigmobile, Aug. 2000, pp. 129–30.

[5] W. Peng and X. Lu, "AHBP: An efficient broadcast protocol for mobile ad hoc networks," *Journal of Computer Science and Technology*, vol. 16, no. 2, pp. 114 – 125, Mar 2001.

[6] S. Basagni, I. Chlamtac, V. R. Syrotiuk, and B. A. Woodward, "A distance routing effect algorithm for mobility (DREAM)," in *MobiCom'98*, New York, NY, USA, Oct. 1998, IEEE, pp. 76–84, ACM.

[7] G. Karumanchi, S. Muralidharan, and R. Prakash, "Information dissemination in partitionable mobile ad hoc networks," in *Proc. 18th IEEE Symp. Reliable Distributed Systems*, Oct. 1999, pp. 4–13.

[8] I. Stojmenovic, M. Seddigh, and J. Zunic, "Dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 13, no. 1, pp. 14–25, Jan. 2002.

[9] R.G. Ogier, F.L. Templin, B. Bellur, and M.G. Lewis, "Topology broadcast based on reverse-path forwarding (TBRPF)," IETF Internet draft, draft-ietf-manet-tbrpf-05.txt, Mar 2001.

[10] L. Bajaj, M. Takai, R. Ahuja, K. Tang, R. Bagrodia, and M. Gerla, "GloMoSim: A scalable network simulation environment," Tech. Rep. 990027, UCLA Computer Science Department, 1999.